In the United States Patent and Trademark Office

In re the application of: Palliyil	)		
Filed: 12/12/2003	)	Group Art Unit:	2139
For: Apparatus, methods and computer programs for identifying matching resources within a data processing network	)	Examiner:	James R. Turchen
Appl. No.: 10/735,509	)		
Appellant's Docket: JP920030270US1	)		

Amended Appeal Brief

This Amended Appeal Brief is responsive to a Notification of Non-Compliant Appeal Brief of September 23, 2008, in which it is asserted that the brief does not contain the items required under 37 CFR 41.37(c), or the items are not under proper heading or in the proper order, and that the brief does not contain a correct copy of the appealed claims in an appendix thereto under 37 CFR 41.37(c)(1)(viii))). The Patent Appeal Center Specialist noted, "Please refer to MPEP CFR 41.37 for proper headings. Section VIII Claims appendix contains marked up claims. Only a clean copy is acceptable."

Please substitute this entire Amended Appeal Brief for Appellant's Appeal Brief previously filed on September 11, 2008.

REAL PARTY IN INTEREST

The assignee, International Business Machines Corporation, is the real party in interest.

RELATED APPEALS AND INTERFERENCES

This is the first appeal in the present patent application. There are no related appeals or interferences known to the appellant or its legal representative.

STATUS OF CLAIMS

Claims pending:

Independent claims 24, 31, and 38 and dependent claims 25-30, 32-37, and 39-44 stand pending in the application as of the time of the Office action herein appealed, (the non-final Office action of March 13, 2008.

Claims previously canceled:

Claims 1-23 were previously canceled.

Claims rejected:

All the pending claims stand rejected in the non-final Office action of March 13, 2008.

Claims appealed:

Claims 24, 31, and 38 are appealed and argued herein.

STATUS OF AMENDMENTS

There are no amendments in connection with this appeal.

The present application, filed December 12, 2003, presented original claims 1-23.

A first, nonfinal Office action of April 3, 2007, rejected all claims under 35 U.S.C. 103(a). In order to overcome the rejections, Appellant responsively canceled claims 1-23 and submitted new claims 24-44.

A second, final Office action of October 17, 2007 rejected all claims. Appellant filed a Request for Continued Examination, accompanied by an amendment which amended independent claims 24, 31 and 38.

The present, nonfinal Office Action of March 13, 2008 rejects all claims.

Appellant appealed the rejection in a Notice of Appeal filed July 11, 2008.

SUMMARY OF CLAIMED SUBJECT MATTER

The present application, as published, provides context for what is claimed, as follows:

[0039] In a local area network environment, it is common for each personal computer 70 to have a similar set of installed computer programs, and for some of the data files stored within the LAN to be replicated across several computers in the network. Therefore, periodic executions of the antivirus software typically involve scanning identical data files and executable files on many different computers. The periodic virus scans involve scanning newly created and newly installed files, but also repeating virus scans of files which were already in existence when the last virus scan was performed. The pre-existing files may not have changed since the last scan, but repeated scanning of pre-existing files has previously been considered essential for protection because timestamps on files cannot be relied on as evidence that the files have not changed.

[0040] The inventors of the present invention have identified these issues as problems requiring a solution. Embodiments of the invention described below use a comparison of hash values computed from the bit patterns representing stored files to identify which files have changed since the last virus scan. The embodiment avoids full virus scanning of files which have not changed since the last scan. Another feature, or alternative embodiment, of the invention also uses a comparison of hash values to identify replicas of files to avoid repetitious virus scanning of multiple replicas. Further embodiments are described thereafter.

. . .

[0097] A further embodiment of the invention uses statistical observation of the pattern of creation of new hashes to identify sudden changes within a network. For example, if newly computed hash values are compared with stored hash values and a large number of copies of a specific hash value MD_1 can be seen to have changed, this implies that the corresponding copies of the resource represented by hash value MD_1 have also changed. This could mean that a group of users are upgrading from one file version to another (for example if MD_1 consistently changes to MD_2) or that a virus is spreading through the system. The latter is most likely if a large number of copies of MD_1 have remained unchanged for a long period and are then suddenly replaced by a large number of different hash values--indicating the probable spread of a polymorphic virus. The comparison of hash values can be used once again to determine which resources require a virus scan and which do not.

Claim 24

Claim 24 describes a method for computing first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored on respective data processing systems within a network. The claim has steps as follows:

Step 1: storing the computed first hash values;

Step 2: computing current hash values for the replicas of the resource;

Step 3: comparing the current and first hash values in order to identify whether all the hash values match, wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value;

Step 4: detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one; and

Step 5: presenting a message for a user indicating a vulnerability, wherein the presenting is responsive to the predetermined number being exceeded.

The specification of the present application provides an exemplary embodiment of the invention and describes the method of claim 24 in terms of that embodiment. Specifically, regarding support for claim 24, see original published application, paragraph 0048, Fig. 2 (computing 200 first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored 220 on respective data processing systems within a network); paragraphs 0048, Fig 2 (storing 220 the computed first hash values); paragraph 0051, Fig. 3 (computing 300 current hash values for the replicas of the resource); paragraph 0051, Fig. 3 (comparing 310 the current and first hash values in order to identify whether all the hash values match 320); paragraph 0045, i.e., "If the file has been modified, a hash value computed after the change will differ from a hash value computed before the change . . ." (wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value); paragraphs 0095, 0097, Fig. 8 (detecting that a vulnerability exists responsive to the hash value comparison 430 indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison 430 indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one); paragraph 0095, Fig. 8 (presenting a message to a user indicating a vulnerability); and paragraph 0097 (wherein the presenting is responsive to the predetermined number being exceeded).

Claim 31

Claim 31 describes an apparatus that includes a processor and a storage device connected to the processor, wherein the processor is operative to execute instructions of the program to implement a method. The claim has steps as follows:

Step 1: computing first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored on respective data processing systems within a network;

Step 2: storing the computed first hash values;

Step 3: computing current hash values for the replicas of the resource;

Step 4: comparing the current and first hash values in order to identify whether all the hash values match, wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value;

Step 5: detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one; and

Step 6: presenting a message for a user indicating a vulnerability, wherein the presenting is responsive to the predetermined number being exceeded.

The specification of the present application provides an exemplary embodiment of the invention and describes the apparatus of claim 31 in terms of that embodiment. Specifically, regarding support for claim 31, see original published application, paragraph 0035 (a storage device connected to the processor, wherein the storage device has stored thereon a program, wherein the processor is operative to execute instructions of the program to implement a method); paragraph 0048, Fig. 2 (computing 200 first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored 220 on respective data processing systems within a network); paragraphs 0048, Fig 2 (storing 220 the computed first hash values); paragraph 0051, Fig. 3 (computing 300 current hash values for the replicas of the resource); paragraph 0051, Fig. 3 (comparing 310 the current and first hash values in order to identify whether all the hash values match 320); paragraph 0045, i.e., "If the file has been modified, a hash value computed after the change will differ from a hash value computed before the change . . ." (wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value); paragraphs 0095, 0097, Fig. 8 (detecting that a vulnerability exists responsive to the hash value comparison 430 indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison 430 indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one); paragraph 0095, Fig. 8 (presenting a message to a user indicating a vulnerability); and paragraph 0097 (wherein the presenting is responsive to the predetermined number being exceeded).

Claim 38

Claim 38 describes a computer program product stored on a tangible, computer readable medium. The computer program product has instructions for execution by a computer system. The instructions, when executed by the computer system, cause the computer system to implement a method. The claim has steps as follows:

Step 1: computing first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored on respective data processing systems within a network;

Step 2: storing the computed first hash values;

Step 3: computing current hash values for the replicas of the resource;

Step 4: comparing the current and first hash values in order to identify whether all the hash values match, wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value;

Step 5: detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one; and

Step 6: presenting a message for a user indicating a vulnerability, wherein the presenting is responsive to the predetermined number being exceeded.

The specification of the present application provides an exemplary embodiment of the invention and describes the computer program product of claim 38 in terms of that embodiment. Specifically, regarding support for claim 38, see original published application, paragraphs 0019, 0035 (A computer program product, stored on a tangible, computer readable medium, said computer program product having instructions for execution by a computer system, wherein the instructions, when executed by the computer system, cause the computer system to implement a method); paragraph 0048, Fig. 2 (computing 200 first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored 220 on respective data processing systems within a network); paragraphs 0048, Fig 2 (storing 220 the computed first hash values); paragraph 0051, Fig. 3 (computing 300 current hash values for the replicas of the resource); paragraph 0051, Fig. 3 (comparing 310 the current and first hash values in order to identify whether all the hash values match 320); paragraph 0045, i.e., "If the file has been modified, a hash value computed after the change will differ from a hash value computed before the change . . ." (wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value); paragraphs 0095, 0097, Fig. 8 (detecting that a vulnerability exists responsive to the hash value comparison 430 indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison 430 indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one); paragraph 0095, Fig. 8 (presenting a message to a user indicating a vulnerability);

and paragraph 0097 (wherein the presenting is responsive to the predetermined number being exceeded).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 24-29, 31-36, and 38-43 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Radatti (US 7,143,113) in view of Szor (US 200510022018).

ARGUMENTS

Rejection Under 35 USC 103(a) 35 U.S.C. 103(a) over Radatti in view of Szor.

Claims 24, 31, and 38

Independent claims 24, 31, and 38 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Radatti in view of Szor. Appellant respectfully submits that the rejection is improper.

Examiner's Position

Claims 24, 31, and 38:

Examiner argues that Radatti discloses computing first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored on respective data processing systems within a network *[Radatti, column 3 lines 77-34, the baseline is formed from the master system, all of the subsequent systems are replicas of the master system; therefore hash values derived from a master system represent a plurality of replicas]*;

a) storing the computed first hash values *[Radatti, column 3 lines 44-48, the secure system data is retained in a storage area, either internally or externally]*;

b) computing current hash values for the replicas of the resource *[Radatti, column 5 lines 28-34, in the comparison cycle, files are taken one at a time and hashed (MD5)*;

c) comparing the current and first hash values in order to identify whether all the hash values match, wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value *[Radatti, column 5 lines 33-58, the recent hash is compared with the old hash];*

d) detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas *[Radatti, column 7 lines 54-58, if an unauthorized user changes the contents, the files modified by the virus will differ];* and

e) presenting a message for a user indicating a vulnerability, wherein the presenting is responsive to the predetermined number being exceeded *[Radatti, column 7 lines 24-28, reporting may be used; as is well known in the art it is inherent that the reporting will take place after detection].*

Examiner acknowledges Radatti does not disclose "the predetermined number is at least one," but argues that Szor is similar to Radatti in that Szor provides a method for network intrusion detection. Examiner observes that Szor discloses a local analysis center (LAC) that receives notification packets about malicious code *[Szor, paragraphs 102-104]*, and that the LAC checks to see if an attack threshold has been exceeded which is incremented by one for each notification packet *[Szor, paragraphs 108-109]*, and then appropriate action is taken *[Szor, paragraph 113]*. Based on this, Examiner argues that it would have been obvious to one of ordinary skill in the art at the time of invention "to modify the method of Radatti to include the functionality of the LAC of Szor in order to determine a minimum level of suspicious activity," citing Szor, paragraph 108.

<u>Appellant's Rebuttal</u>

Claims 24, 31, and 38:

Appellant respectfully submits that amended claims 24, 31, and 38 are patentably distinct because the references relied upon do not teach or suggest all of what the claims recite. MPEP 2143.03 ("All words in a claim must be considered in judging the patentability of that claim against the prior art," citing In re Wilson, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970)). Further, no rational underpinning has been stated for modifying the references such that all of what the claims recite would have been obvious to one of ordinary skill in the art at the time of invention. MPEP 2143.01 III (citing KSR International Co. v. Teleflex Inc., USPQ2d 1385, 1396 (U.S.S.C. 2007)).

Claims 24, 31, and 38 indicate first hash values, derived from and representing a plurality of replicas of a resource (e.g., a file) are computed and stored. Current hash values for the replicas of the resource are computed. Further, claims 24, 31, and 38 go on to state that the current and first hash values are compared "in order to identify whether all the hash values match." Still further, independent claims 24, 31, and 38 recite "detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one." In other words, more than one file must be changed in order to indicate a vulnerability.

Even aside from the amendments submitted herein, the cited references do not teach or suggest the claimed combination of features. In particular, the primary reference, Radatti, actually *teaches away* from what is claimed in at least one respect. That is, Radatti teaches that *any* difference in hash value comparison indicates a vulnerability, whereas claims 24, 31, and 38 make it clear that a vulnerability is indicated only if there is *more than one* changed replica (as indicated by differences in first and current hash values of the respective replicas). For example, Radatti, col. 7, lines 47-58, teaches that the system responds to even a single change in a file, by stating that ". . . *any* variation from the secure system state will be detected. The nature of the file will not matter insofar as *any file* that modifies the system and/or its files will be detected" (emphasis added). This is different than detecting a vulnerability responsive only to a plurality of changes in a replica (e.g., a file), i.e., a "detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one," as claimed in the present case.

The rejection relies upon Szor for the teaching that "the predetermined number is at least one" in analysis done by a local analysis center ("LAC"). Szor teaches monitoring activity of malicious fileless code. Szor, paragraph 37. This is done in order to detect the code before signatures have been developed for it. Szor, paragraph 10. An example is monitoring calls to a critical "send" function that is provided by an operating system. Szor, paragraphs 38-39. Szor teaches that when such a send call occurs and it is determined by checking operation 204 to *not* be an attack, or at least a suspected attack, a behavior blocking application 126 merely continues to check for attacks at 204. Szor, FIG. 2 and paragraph 36.

In the teaching of Szor, once checking operation 204 determines that a send call *is* at least a suspected attack, then counting may arise wherein application 126 continues to a next checking operation 206 and ultimately may continue to an operation 222 in which malicious fileless code, or a snippet of the fileless code, is sent to the LAC in a packet. Szor, FIG. 2 and paragraph 101. The LAC may determine conclusively at operation 408 that a single instance of certain received code is an attack. Szor, FIG. 4 and paragraph 110. However, the LAC might *not* determine this conclusively based merely upon a single instance of certain received code, in which case the LAC must receive some number of multiple instances of such code (which indicates multiple instances of activity, i.e., a send call, due to the code) before the LAC concludes the code's activity represents an attack. Szor, FIG. 4 and paragraph 111.

The "predetermined number" recited by the claim is a predetermined number of changed *replicas of a resource.* As indicated in the above discussion, Szor does not teach that in analysis done by the LAC, the LAC finds that the predetermined number *of changed replicas of a resource* is at least one. Thus the rejection relies not simply upon modifying the teachings of Radatti "to include the functionality of the LAC of Szor." The rejection also relies on modifying the teaching of Szor regarding the functionality of the LAC, such that in analysis done by the LAC, the LAC finds that the predetermined number *of changed replicas of a resource* is at least one.

The rejection must state a rational underpinning for the modification of both the teaching of Radatti and Szor. MPEP 2143.01 III (citing KSR International Co. v. Teleflex Inc., USPQ2d 1385, 1396 (U.S.S.C. 2007)). This has not been done. That is, the rejection modifies *determining a minimum suspicious level*[1] *by detecting even merely one change*, as taught by Radatti, to adopt Szor's teaching of *determining a minimum suspicious level by detecting*

[1] The claim in the present application states this as "detecting that a vulnerability exists."

more than one instance. The stated rationale for the modification is "in order to determine a minimum suspicious level of activity." Appellant respectfully submits that this is mere circular logic and not a rational underpinning for the modification.

Further, the rejection modifies the teaching of Szor. The need to modify Szor in order to apply it to the claim of the present invention arises at least partly because the problem addressed by Szor is not precisely the same as those addressed in Radatti and in the present invention. This tends to indicate, in and of itself, that the propounded combination and modification should be subject to more scrutiny than if the problems are precisely the same. Compare to MPEP 2143.01 I (citing Ruiz v. A.B. Chance Co., 69 USPQ2d 1686, 1690 (Fed. Cir. 2004), in which motivation was found sufficient at least partly because the references dealt with *precisely the same problem*).

Szor teaches a host computer monitoring calls to a critical function of an operation system and responsively sending fileless code to an LAC if the code called that function and is deemed suspicious by an attack checking operation 204, and then the LAC counting instances of received suspicious fileless code (or portions thereof), as explained above. Contrast this to "detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource" [or "file," if the amendment submitted herein is entered], where hash values represent "replicas . . . stored on respective data processing systems," as claimed, and as the rejection contends that Radatti teaches.

In order to modify the LAC of Szor so that the LAC finds that a predetermined number *of changed replicas of a resource* is at least one, requires that the LAC would have to be modified to also compare instances of received suspicious fileless code to see if they are changed versions of an original instance, which would also require that the LAC receive information to enable it to do this. There is no suggestion of such modification in the references and no rationale is given in the rejection for such a modification. Appellant respectfully submits that the rejection is, therefore, improper for this reason, as well.

Further, the proposed modification cannot change the principle of operation of a reference. MPEP 2143.01 VI (citing In re Ratti, 123 USPQ 349 (CCPA 1959). But the principle of operation of LAC in Szor is substantially different than that of Radatti, as explained above. Appellant submits that it would impermissibly change the principle of operation of Szor to modify the LAC to also compare instances of received suspicious fileless code to see if they are changed versions of an original instance, and to modify the host so that the information sent to the LAC enables the LAC to do this. Appellant respectfully submits that the rejection is, therefore, improper for this reason, as well.

REQUEST FOR ACTION

For the above reasons, Appellant contends the invention defined in claims 24, 31, and 38 is patentably distinct. Appellant requests that the Board grant allowance and prompt passage of the application to issuance.

Respectfully submitted,

By Anthony V S England

Anthony V.S. England
Registration No. 35,129
Attorney of Record for
IBM Corporation
Telephone: 512-477-7165
a@aengland.com

Attachments: Claims Appendix, Evidence Appendix, Related Proceedings Appendix

APPENDIX "AA" CLAIMS

1-23. (canceled)

24. (previously presented) A method comprising the steps of:

computing first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored on respective data processing systems within a network;

a) storing the computed first hash values;

b) computing current hash values for the replicas of the resource;

c) comparing the current and first hash values in order to identify whether all the hash values match, wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value;

d) detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one; and

e) presenting a message for a user indicating a vulnerability, wherein the presenting is responsive to the predetermined number being exceeded.

25. (previously presented) The method of claim 24, wherein steps a), b), c), and d) are performed at a first data processing system within the network.

26. (previously presented) The method of claim 24, wherein step b) is performed at each replica's respective data processing system, the method further comprising sending the computed hash values to a first data processing system.

27. (previously presented) The method of claim 24, wherein the vulnerability includes a vulnerability to a computer virus.

APPENDIX "AA" CLAIMS

28. (previously presented) The method of claim 24, wherein the vulnerability includes a vulnerability to computer hacking.

29. (previously presented) The method of claim 24 further comprising:

classifying as vulnerable the data processing systems storing the replicas, wherein the classifying is responsive to the predetermined number of changed replicas of the resource being exceeded.

30. (previously presented) The method of claim 24, the steps further comprising:

sending a notification of the vulnerability to each data processing system storing one of the replicas;

selecting a sequence of vulnerability-resolution instructions relevant to the vulnerability; and

sending the selected instructions to each of the data processing systems storing one of the replicas.

APPENDIX "AA" CLAIMS

31. (previously presented) An apparatus comprising:

a processor; and

a storage device connected to the processor, wherein the storage device has stored thereon a program, wherein the processor is operative to execute instructions of the program to implement a method comprising the steps of:

computing first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored on respective data processing systems within a network;

a) storing the computed first hash values;

b) computing current hash values for the replicas of the resource;

c) comparing the current and first hash values in order to identify whether all the hash values match, wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value;

d) detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one; and

e) presenting a message for a user indicating a vulnerability, wherein the presenting is responsive to the predetermined number being exceeded.

32. (previously presented) The apparatus of claim 31, wherein steps a), b), c), and d) are performed at a first data processing system within the network.

33. (previously presented) The apparatus of claim 31, wherein step b) is performed at each replica's at respective data processing system, the method further comprising sending the computed hash values to a first data processing system.

APPENDIX "AA" CLAIMS

34. (previously presented) The apparatus of claim 31, wherein the vulnerability includes a vulnerability to a computer virus.

35. (previously presented) The apparatus of claim 31, wherein the vulnerability includes a vulnerability to computer hacking.

36. (previously presented) The apparatus of claim 31, the steps further comprising:

classifying as vulnerable the data processing systems storing the replicas, wherein the classifying is responsive to the predetermined number of changed replicas of the resource being exceeded.

37. (previously presented) The apparatus of claim 31, the steps further comprising:

sending a notification of the vulnerability to each data processing system storing one of the replicas;

selecting a sequence of vulnerability-resolution instructions relevant to the vulnerability; and

sending the selected instructions to each of the data processing systems storing one of the replicas.

APPENDIX "AA" CLAIMS

38. (previously presented) A computer program product, stored on a tangible, computer readable medium, said computer program product having instructions for execution by a computer system, wherein the instructions, when executed by the computer system, cause the computer system to implement a method comprising the steps of:

computing first hash values derived from and representing a plurality of replicas of a resource, wherein the replicas are stored on respective data processing systems within a network;

a) storing the computed first hash values;

b) computing current hash values for the replicas of the resource;

c) comparing the current and first hash values in order to identify whether all the hash values match, wherein nonmatching first and current hash values for a respective one of the replicas indicates the respective one of the replica has changed since the computing of the first hash value;

d) detecting that a vulnerability exists responsive to the hash value comparison indicating more than a predetermined number of changed replicas of the resource, and that no vulnerability exists responsive to the hash value comparison indicating less than or equal to the predetermined number of changed replicas, wherein the predetermined number is at least one; and

e) presenting a message for a user indicating a vulnerability, wherein the presenting is responsive to the predetermined number being exceeded.

39. (previously presented) The computer program product of claim 38, wherein steps a), b), c), and d) are performed at a first data processing system within the network.

40. (previously presented) The computer program product of claim 38, wherein step b) is performed at each replica's respective data processing system, the method further comprising sending the computed hash values to a first data processing system.

APPENDIX "AA" CLAIMS

41. (previously presented) The computer program product of claim 38, wherein the vulnerability includes a vulnerability to a computer virus.

42. (previously presented) The computer program product of claim 38, wherein the vulnerability includes a vulnerability to computer hacking.

43. (previously presented) The computer program product of claim 38, the steps further comprising:

classifying as vulnerable the data processing systems storing the replicas, wherein the classifying is responsive to the predetermined number of changed replicas of the resource being exceeded.

44. (previously presented) The computer program product of claim 38, the steps further comprising:

sending a notification of the vulnerability to each data processing system storing one of the replicas;

selecting a sequence of vulnerability-resolution instructions relevant to the vulnerability; and

sending the selected instructions to each of the data processing systems storing one of the replicas.

APPENDIX "BB" EVIDENCE

NONE.

APPENDIX "CC" RELATED PROCEEDINGS

NONE.